View metadata, citation and similar papers at core.ac.uk
brought to you by CORE
provided by DBIS EPub

Monitoring Dependencies for SLAs: The MoDe4SLA Approach

Lianne Bodenstaff[1]*, Andreas Wombacher[2]
IS Group[1], DB Group [2]
University of Twente
The Netherlands
{l.bodenstaff|a.wombacher}@utwente.nl

Manfred Reichert
DBIS Group
University of Ulm
Germany
manfred.reichert@uni-ulm.de

Michael C. Jaeger
KBS Group
Techn. Universität Berlin
Germany
mcj@cs.tu-berlin.de

Abstract

*In service oriented computing different techniques for monitoring Service Level Agreements (SLAs) are available. Many of these monitoring approaches focus on bilateral agreements between partners. However, when monitoring composite services it is not only important to figure out whether SLAs are violated, but we also need to analyze why these violations have occurred. When offering a composite service a company depends on its content providers to meet the service level they agreed upon. Due to these dependencies a company should not only monitor the SLA of the composite service, but also the SLAs of the services it depends on. By analyzing and monitoring the composite service in this way, causes for SLA violations can be easier found. In this paper we demonstrate how to analyze SLAs during development phase and how to monitor these dependencies using event logs during runtime. We call our approach MoDe4SLA (***Mo****nitoring* ***De****pendencies for SLAs).*

1. Introduction

For a business operating in a networked environment it is vital to accurately manage the services it provides to its customers. This is particularly challenging when a business offers *composite* services where interaction with services offered by other providers influence its performance. The quality of service, which can be offered to customers, is calculated taking all these dependencies into account [10], [7]. Consider a composite service which returns combined information from several search engines. In this case, the quality of service (e.g. response time) which can be offered depends on the quality of service delivered by the search engines. Together with constraints of the customer, these calculations form the basis for a Service Level Agreement (SLA) between customer and service provider [12], [14]. Several approaches exist for *monitoring* the service level of services during runtime (e.g. [15]). The monitoring results are compared with the constraints specified in the SLA for possible violation detection. Since SLAs are typically bilateral agreements, current monitoring approaches focus on identifying violations in bilateral communication. Important research questions in this area are, for example, how to gather reliable data, how to structure these data, and how to combine data from different sources. Such monitors are often process specific, activated or created when a service is invoked, and terminated when the service is completed.

Most approaches for managing composite services combine the level of quality a company can provide with monitoring bilateral communication. However, to properly manage its composite service a company has to be able to reason about *causes* of SLA violations. For example, when the offered response time for a composite service provided by a company depends on response times of other services this company uses, it is vital to identify and monitor these *dependencies*. Exactly these dependencies are ignored by bilateral monitoring approaches. However, combining bilateral monitoring results based on their dependencies is, as we discuss in this paper, highly challenging. With the MoDe4SLA approach we analyze during development phase different types of dependencies between services, and the *impact* services have on each other. Further, it allows to combine bilateral monitoring results with analyzed dependencies and impacts of the composite service.

We demonstrate our approach by means of an intuitive example in which a company offers two composite services to its customers. *SubscribedNews* is a composite service where customers automatically receive a news report. This report is created by combining news items on personal interest (e.g., stock market information) from two different news providers. Customers pay a flat fee for this service. *NewsRequest*, in turn, is a composite service which allows customers to request up-to-date news information on a specific

*This research has been supported by the Dutch Organization for Scientific Research (NWO) under contract number 612.063.409

search query. The NewsRequest service is paid per invocation. For every request the company invokes two content providers and sends information from the fastest responding to the customer.

To offer the SubscribedNews service to its customers, the company automatically receives data (i.e., news items) from its content providers (i.e., news providers). This data is stored in a *database* from which the company retrieves data when composing the news report for its customers. The company has an SLA with both content providers and customers. Furthermore, providing the news report for customers does not trigger a service invocation by the company to the content providers for data, since up-to-date data is retrieved from the database. Therefore, obtaining data from content providers and sending reports to customers are different processes (i.e., invocations). However, response time performance for offering NewsRequest is highly dependent on the response time of the service provided by the content providers. As a result, SLA violations between the company and its providers might result in SLA violations from company to customer, which cannot be identified using a bilateral monitoring approach. For example, when the content provider never meets the response time constraint, the company most likely will also not be able to meet the response time it agreed upont with the customer. Therefore, it is highly important for a company to not only monitor SLA violations for each metric but also their dependencies on the same metrics in other SLAs to identify *causes*.

Our MoDe4SLA approach allows to monitor these dependencies between SLAs, enabling decision support when managing composite services. Fig. 1 gives an overview of MoDe4SLA. Our contribution is to provide an approach which allows the company to analyze its SLAs (annotated with 1 in Fig. 1) by:

- identifying relevant *dependency* relations between services during development phase (annotated with 2 in Fig. 1) (cf. Sect. 3),
- analyzing the *impact* these dependent services have on the composite service during development phase (annotated with 3 in Fig. 1) (cf. Sect. 4),
- *structuring* monitoring results based on dependency relations at runtime (annotated with 4 in Fig. 1) (cf. Sect. 5.1), and
- identifying causes for SLA violations by graphically representing the *comparison* of development phase dependency analysis with runtime monitoring results (annotated with 5 in Fig. 1) (cf. Sect. 5.2).

2. MoDe4SLA Approach

Our approach aims at supporting a company in managing its composite services by identifying and monitoring their

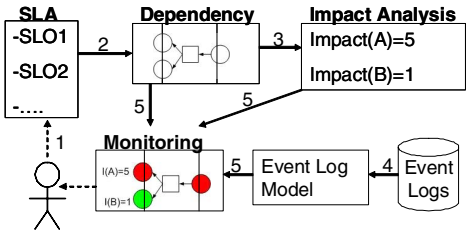


Figure 1. MoDe4SLA Approach

dependencies to services requested from other providers. These dependencies are represented in a *dependency model*. For each constraint in the SLA of a composite service, the services it depends on are identified. An SLA typically consists of a set of *Service Level Objectives* (SLO) which contain guaranteed quality constraints [12], [14]. A typical example of an SLO is:

> *In 90% of all cases, invocation of service X will have a response time within y milliseconds (ms).*

Each of these SLOs is *measurable* and consists out of one or more, possibly composite, *metrics*. For example, a composite metrics can be the average response time over all customers in a month. Furthermore, these SLOs typically hold a validity *time constraint*, for example "Mo-Fri 9:00-17:00".

Offering a composite service to customers implies that a company relies on content providers to offer necessary data. Both customers and content providers have an SLA with the company. We propose to explicate for each SLO on which other services its performance *depends* (cf. Fig. 1). Different SLOs in one SLA might depend on different services or depend on them in different ways. For example, if a company offers information with a fast response time by querying five providers and returning information of the fastest responding one, a cost constraint will be influenced by all five services (due to invocation they all have to be paid) while a response time constraint will only be influenced by the fastest responding service.

Furthermore, we demonstrate how to calculate the *impact* a service has on a depending composite service. Assume for example that the cost constraint on a composite service depends on service A and service B, where service A costs on average ten times more than service B. Now, the cost impact of service A on the composite service is ten times higher than the one of service B. Based on the dependency model we calculate the impact of a service on the composite service (cf. Fig. 1), using impact analysis.

Data on messages exchanged between customer and provider are gathered in *event logs*. These event logs enable monitoring the SLOs and their dependencies based on

SLA *SubscribedNews*, Company & Customer. January 1, 2008 - June 30, 2008
SLO-cost: Company will deliver SubscribedNews on a monthly average of less than 1 euro per service. SLO-time: Company will deliver SubscribedNews once a day before 9:00 am, Mo-Fri.
SLA *NewsUpdate*, Company & CP1. January 1, 2008 - June 30, 2008
SLO-cost: CP1 will deliver NewsUpdate for 0.50 euro per news item, with a maximum of 50 euro per day. SLO-time: CP1 will deliver NewsUpdate to Company once a day, 7 days a week before 6:00 am.
SLA *NewsUpdate*, Company & CP2. January 1, 2008 - June 30, 2008
SLO-cost: CP2 will deliver NewsUpdate for 0.30 euro per news item, with a maximum of 39 euro per day. SLO-time: CP2 will deliver NewsUpdate to Company once a day, 7 days a week before 6:00 am.

Table 1. SLAs SubcribedNews

SLA *NewsRequest*, Company & Customer. January 1, 2008 - June 30, 2008
SLO-cost: Invocation NewsRequest by Customer will cost the first 100 times in a month 0.50 euro, thereafter 0.40 euro. SLO-time: Company will respond to NewsRequest invocation by Customer within 5 ms, 99% of the time.
SLA *Request*, Company & CP1. January 1, 2008 - June 30, 2008
SLO-cost: Invocation Request by Company will cost 0.20 euro. SLO-time: CP1 will respond to Request invocation by Company within 3 ms, 99.9% of the time.
SLA *Request*, Company & CP2. January 1, 2008 - June 30, 2008
SLO-cost: Invocation Request by Company will cost 0.15 euro. SLO-time: CP2 will respond to Request invocation by Company within 4 ms, 99% of the time.

Table 2. SLAs NewsRequest

these data. We abstract and structure necessary data from event logs in an *event log model* to enable evaluation and monitoring of SLOs (cf. Fig. 1). For decision support in managing composite services, we combine the dependencies, the calculated impact factors, and the bilateral monitoring results into one model which graphically represents these relations.

3. Dependency Models

A composite service depends on one or more other services. In our example, SubscribedNews and NewsRequest both depend on two other services offered by different content providers. In our approach, we specify on *which* services the fulfillment of a specific SLO depends and *how* it depends on these services. The SLOs for cost and response time, specified in the SLAs of the company with content providers (CP1 and CP2) and customers are denoted in Table 1 and Table 2. Note that we do not provide a specification language for SLAs like WSLA Framework [12] or the SLA language by Sahai et al. [14]. Instead our approach focusses on conceptual issues and is thus language independent.

The company delivers from Monday to Friday before 9:00 am news items the customer is interested in. The price is not higher than 1 euro and depends mainly on the number of news items that fit the requirements of the customer. With its content providers the company agreed to deliver daily all news items for a fixed price per item with a maximum of, respectively, 50 and 39 euro per day. These news items are delivered before 6:00 am. Regarding the NewsRequest service the customer can request news on a specific topic. These requests costs 0.50 euro. If the customer has more than 100 requests per month, price per invocation will decrease to 0.40 euro. Each request will be in 99% of all cases responded within 5 ms. The company, in turn, invokes the services of two content providers for every NewsRequest and sends data from the fastest responding to its customer. Content providers respond within 3 ms (4 ms) for 0.20 euro (0.15 euro) to an invocation by the company in 99.9% (99%) of all cases.

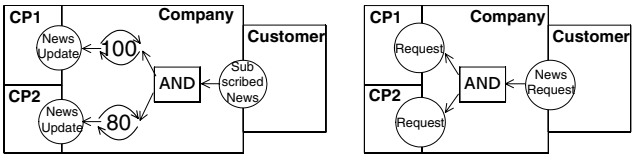
Although dependency models for SLOs might differ in notation, common requirements can be identified:

- Req. 1: Model the composite service and the services it depends on.
- Req. 2: Model dependency on more than one service.
- Req. 3: Model dependency on one service out of a set. Quantification has to be possible.
- Req. 4: Model storage with or without reuse.

Req. 2 ensures that the cost of a composite service may depend on the costs of two other services. Req. 3 states that when a choice between services occurs, the company should be able to estimate how often each of the services will be chosen. Req. 4 enables to model reuse of data from a service. For example, in SubscribedNews the company uses a database of information to serve its customers and it fills the database by querying several content providers. Data from content providers can be *reused* in other composite services. In Sect. 3.1 and Sect. 3.2 the construction of dependency models for the SLOs of the composite services SubscribedNews and NewsRequest are described.

#	Construct	Explanation
1	A → B	Service A depends on service B
2	AND	AND-split/join
3	XOR	XOR-split/join
4	X	Reuse is X times

Table 3. Cost Model Constructs



(a) SubscribedNews (b) NewsRequest

Figure 2. Cost Dependency Model

3.1. Dependency Model for Cost

Both examples (i.e., SubscribedNews and NewsRequest) contain one SLO depicting a *cost* constraint (cf. Table 1 and 2). Here, we construct for both examples the *cost dependency model*. We introduce a domain-specific modelling language for cost dependencies which enables depicting all necessary details while keeping a high level of abstraction. This enhances readability and decreases necessary modelling effort. The requirements enumerated in Sect. 3 are implemented by the constructs from Table 3. It is possible to model that meeting a constraint for one service *depends* on how the performance of another service is. It is also possible to model that meeting constraints for one service depends on two or more different services with an *AND-split*. Furthermore, by using *XOR-splits* exclusive choices can be modelled (i.e., the service depends on one out of a set of services) with the possibility to add a *ratio* on the likelihood of choosing one of the options. As a last modelling construct it is possible to denote *reuse* of data provided by a service.

Fig. 2 shows on *which* services the cost of SubscribedNews and NewsRequest depend and *how* these dependencies look like. The cost of composite service SubscribedNews is dependent on the cost of NewsUpdate with both CP1 and CP2, indicated with an AND-split (cf. Fig. 2). Moreover, NewsUpdate data is used more than once (i.e. the company sends more than once the same data to different customers). This *reuse* is 100 times of data from CP1 and 80 times from CP2 (cf. Fig. 2). Note that data from CP1 are more often reused than data from CP2 because the former are also used in another unrelated service offered by the company.

The cost of composite service NewsRequest is dependent on both costs of Request from CP1 and CP2. This is again indicated with an AND-split (cf. Fig. 2). So, even though the customer only receives data from one service provider, he pays for fast delivery, namely the cost of invoking two content providers. Data for a news request are not reused.

3.2. Dependency Model for Response Time

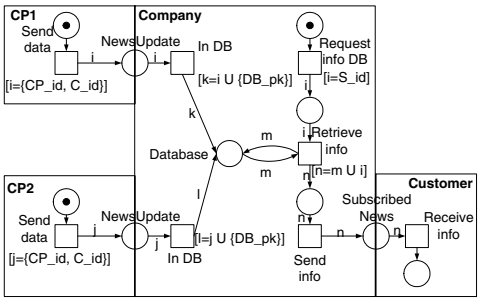
SubscribedNews and NewsRequest both contain an SLO with a *time* constraint (cf. Table 1 and 2). Next, we introduce the domain specific *response time dependency model* for both examples. Apart from the general modelling requirements for dependency models as depicted in Sect. 3, also the *order* in which services are executed is relevant for response times. For example, if the services on which the composite service depends can be executed in parallel, the response time will be faster than when compared to serialized execution. Workflow modelling techniques are highly suitable for modelling response time dependencies, allowing ordering of messages. We use Coloured Petri Nets (CPN) [11] for these models. Table 4 discusses the requirements as summarized in Sect. 3 for Petri Nets. We illustrate how the different requirements can be modelled. Note that by using Petri Nets these requirements can be modelled in many different ways.

#	Construct	Explanation
1	A → B	Service B depends on service A. Also: serial execution of services.
2		AND-split: parallel execution.
3		XOR-split.
4	DB	Database, DB, with reuse of data.

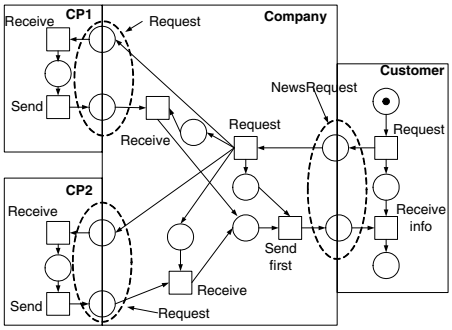
Table 4. Response Time Constructs

The response time for SubscribedNews is dependent on NewsUpdate services of both CP1 and CP2 (cf. Fig. 3). The CPN depicts how CP1 and CP2 send data which are then stored in the database of the company. Both content providers send their id (`CP_id`) together with the content (i.e., news items, `C_id`). The company adds this information to the database while inserting a primary key (DB_pk). Independent from filling the database, the company sends daily updates to customers with specific interests by requesting data from the database (`S_id`). Data used to compose a news report for the customer can be reused for other reports. Therefore, information requested from the database are also returned (m). The resulting report is sent to the customer.

Response time dependencies for NewsRequest are dependent on the response times of Request services invoked



(a) SubscribedNews



(b) NewsRequest

Figure 3. Time Dependency Models

by the company to both content providers (cf. Fig. 3). For readability purposes we omit coloring of the Petri Net. As soon as the company receives a response from one of the two content providers, it sends out this information to the customer (Send first). This enables faster response times for the customer.

4. Impact Analysis

The dependency model for an SLO depicts on which services a composite service depends. However, not every service the composite service depends on, has the same *impact* on the composite service. For example, if costs for a composite service depend on services A and B, but service A costs on average only ten percent of service B, the impact of service B on the price of the composite service is much higher than the impact of service A. Therefore, our approach allows to do an *impact analysis* for every dependency model. The impact a service has on the composite service can be determined by analyzing the dependency model. Each construct (e.g. AND-split) in the dependency model affects the impact of the service it connects to the composite service in a different way. For example, an AND-split in a cost dependency model denotes that costs for a composite service are influenced by both services it depends on. Opposed to this, an XOR-split indicates that the costs

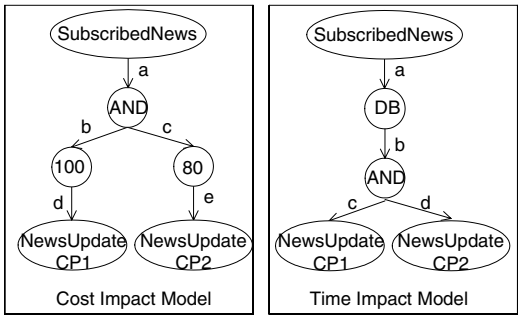
Graph	Equation	Cost	Response Time
x → A^r → y	$I(A) = xAr$ $y = x$	A	A
x → AND → y, z	$y = x$ $x = z$	AND	All parallel splits and joins
x → XOR → r: y, s: z	$y = rx$ $z = sx$	XOR	All exclusive choices
x → OR → r: y, s: z	$y = rx$ $z = sx$		All m-out-of-n choices
x → DB → y	$y = 0$		All databases and storages
x → RE → y	$y = \frac{x}{RE}$	X	

Table 5. Nodes of Abstraction Graph

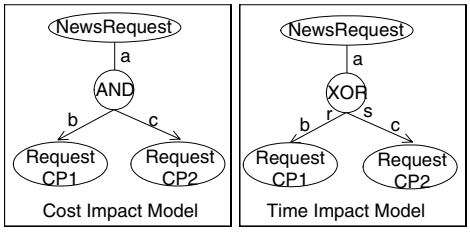
for the composite service is influenced by only one of the two services it depends on (i.e., a service only has an impact on part of the composite service instances), which decreases the impact of these services with 50% (i.e., each service only occurs in half of the composite service instances). This impact factor indicates how likely it is that the service influenced the performance of the composite service.

To calculate this impact factor, we need to go through the dependency model, starting at the composite service through all constructs, ending at the services. By considering the effects of each construct on the impact while going through the model, we can calculate the impact of the service. For analyzing the dependency model in a structured way we create an *abstraction graph* to depict the impact models. For each construct in the dependency model a node with its impact equation is created. Table 5 denotes the *graph notation* and *equation* for all constructs of cost and response time dependency models. The comparison impact value of the composite service is always 1.

The impact factor of service A ($I(A)$) on a composite service is equal to factor x on the incoming edge times the value of service A times number of loops r (cf. Table 5). Value A is either the estimated cost or the estimated response time. The number of loops r will be important if one service gets invoked more than once for one composite service. For example, if for a composite service the company invokes twice ($r = 2$) the same service A of its content provider with a response time of 3 ms before responding to a request of its customer. If service A results in the invocation of another service (serialism) the outgoing edge (y) will have the same impact factor as the incoming one (x) (cf. Table 5). The second construct is an *AND-split* where outgoing edges (y and z) have the same impact factor as the incoming edge (x). The third construct is an *XOR-split* where estimated ratio r:s determines the impact of edges y and z. Assume that for fulfilling a composite service a company



(a) SubscribedNews



(b) NewsRequest

Figure 4. Impact Models

chooses between two content providers. The first offers fast, but expensive data, the second offers slow but cheap data. Depending on the customer, the company chooses either of the two. Estimations by the company are that it will choose 3:1 (r:s) for the fast service. The fourth construct is an *OR-split* to enable modelling response time dependencies on a *subset* of a set of services (i.e., m-out-of-n services). For example, choosing the first three responses on ten invocations. The equation is the same as for the XOR equation except that $r+s \geq 1$ holds. The fifth construct enables modelling a *database* or *storage*. When the results of a service are stored in a database or, with physical goods, in a storage then the dependency on response time for the composite service will disappear. Therefore, the impact of services adding information or goods to a database or storage on the composite service regarding response time is zero. The last construct in Table 5 enables modelling reuse which influences the cost impact factor. When information is reused the cost for the service is divided among these services. Therefore, the impact factor for costs is divided by the estimated number of reuses.

To ensure completeness for our impact model constructs we analyzed all Workflow Patterns [1]. The influence on the response time for each of these patterns is evaluated and can be represented with the constructs of the abstraction graph. Fig. 4 depicts impact graphs for both response time and cost of the SubscribedNews service as abstracted from the dependency model. For *cost* the service depends on both UpdateNews services with a reuse element in between, to mitigate the influence. Fig. 4 also depicts the impact graphs for *response time* and *cost* of the NewsRequest service. The impact factor for response time is now influenced by the estimated ratio r:s on the XOR-split. Using the equations in Table 5 and both graphs in Fig. 4 cost and time impact factors can be calculated and are depicted in Table 6. Note that the average cost of NewsUpdateCP1 cannot directly be derived from the SLO (cf. Table 1) since it states 0.50 euro per item with a maximum price of 50 euro per day. The same holds for NewsUpdateCP2. Now, the company makes an estimation on the average value based on previous observations: NewsUpdateCP1 $= 40$ and NewsUpdateCP2 $= 30$. For the impact on response time for NewsRequest estimations are made on the XOR-split ratio r:s. Since RequestCP1 should respond on average faster than RequestCP2, ratio r:s is estimated on 3:1. Recall that the agreed upon response times by CP1 and CP2 for Request are 3 and 4 ms. The impact factor is an absolute value, i.e., when an impact value of a service is 5 in Model A and 10 in Model B then the impact of that service on the composite service is really twice as big in Model B as Model A.

Impact Factor	Equation	Result
$I\_cost(NewsUpdateCP1)$	$\frac{1}{100}40$	0.4
$I\_cost(NewsUpdateCP2)$	$\frac{1}{80}30$	0.375
$I\_time(NewsUpdateCP1)$	0	0
$I\_time(NewsUpdateCP2)$	0	0
$I\_cost(RequestCP1)$	0.20	0.20
$I\_cost(RequestCP2)$	0.15	0.15
$I\_time(RequestCP1)$	$3\ times\ 3$	9
$I\_time(RequestCP2)$	$1\ times\ 4$	4

Table 6. Impact Factors of Examples

5. Monitoring

To manage composite services it is important for a company to monitor and evaluate the dependency models. For example, identifying constant violations of an SLO by a provider may lead to ending collaboration with this partner. On the other hand, when the company can easily meet the agreed service level, it might be able to make a more competitive offer with a higher level of service. Especially, monitoring dependencies gives insight into the causes for under or over performance of an SLO, supporting decision making on service level changes. Combining this information with impact factors of each service and bilateral monitoring information results in a complete picture of how the composite services are functioning. First, we describe how to abstract necessary data from event logs which are created for bilateral monitoring in Sect. 5.1. Subsequently, Sect. 5.2 discusses how to combine this information with the impact

factors and the identified dependencies.

5.1. Event Log Model

All data necessary to evaluate the dependency models are present in event logs. These are produced during message exchanges. The challenge is to abstract *useful* information and to *structure* it in a meaningful way. The focus of our monitoring approach is on the different SLOs of a composite service. Therefore, we capture information concerning the SLO (e.g., timestamps to measure response times and payment information for cost calculations). In addition we capture data to correlate different messages. A more detailed discussion on how to structure Event Log Models can be found in [6]. To enable abstraction of event log data, we need to correlate the following information:

1. Messages exchanged between provider and customer during invocation of the composite service.
2. Services (and their messages) invoked by the composite service, i.e., the services a composite service depends on. This is achieved by either matching identifiers (as available for example with a BPEL implementation) or by doing semantic matching.
3. Instances of services belonging to one specific type of service (e.g., all instances of a premium account).
4. Grouping classes of services belonging to one business activity (e.g., all premium and normal account instances for service X).
5. Instances of services that are exchanged with the same business partner (e.g., all instances of services exchanged with content provider A).
6. Composite services that reuse data from the same service. Again this information is correlated through identifiers or by semantic matching.

SLOs of one SLA can have different levels of *abstraction*. When one SLO is defined on a lower level (e.g. every invocation of a composite service has a certain response time) while another SLO is defined on a higher level (e.g. all invocations of this composite service over a year have an average cost value of y) then both levels of abstraction are represented in the event log model. For easy representation and implementation, we use sets and tuples. To meet the identified correlation criteria we structure our event log model according to the following definitions. For brevity we denote timestamps in relative ms instead of complete date and time notation.

Definition 1 (Message) *A message x is represented as tuple (a,b,c,d) with time(x)= a, issuer(x)=b, recipient(x)=c, content(x)= d.*

Definition 2 (Service) *A service is represented as a tuple containing:*

- *The set of messages exchanged to execute the service*
- *The cost of the service*

Definition 3 (Composite Service) *A composite service in the event log model is represented as a tuple containing:*

- *The issuer of the composite service*
- *The type of service issued*
- *The cost of the service*
- *The set of messages exchanged with the issuer*
- *The set of service instances this composite service depends on*

An example of one instance of NewsRequest by a customer in event log model notation is as follows.

Composite Service:
CS(id1)=(
CustomerX, NewsRequest, 0.50, {M(id2),M(id3)}, {S(id4), S(id5)})
Services:
S(id4)= (M(id6), M(id7), 0.20)
S(id5)= (M(id8), M(id9), 0.15)
Messages:
M(id2)= (1.0, CustomerX, Company, "Dutch politics")
M(id3)= (5.5, Company, CustomerX, ContentY)
M(id6)= (2.0, Company, CP1, "Dutch politics")
M(id7)= (4.2, CP1, Company, ContentY)
M(id8)= (2.1, Company, CP2, "Dutch politics")
M(id9)= (5.1, CP2, Company, ContentZ)

Here, composite service NewsRequest CS(id1) is issued by CustomerX for the cost of 0.50 euro. Between the company and CustomerX two messages M(id2) and M(id3) were exchanged, and the two services S(id4) and S(id5) were used by the company to fulfill the composite service. Both services S(id4) and S(id5) consist of two messages and cost 0.20 and 0.15 euro, respectively. Each message M(idX) consists of timestamp, issuer, recipient, and content of the message. In this case, the customer wants an update on Dutch politics. Both CP1 and CP2 deliver news items (ContentY and ContentZ). The company sends ContentY since this is the first to arrive (4.2 ms $<$ 5.1 ms).

5.2. Monitoring Dependencies

Analysis of dependencies between services during development phase (cf. Sect. 3), and analysis of impact these services have on the composition (cf. Sect. 4) are used to support monitoring composite services during runtime. The event log model, containing results of bilateral communication, is compared with the SLOs. For example, if the SLO states that response time will be on average 3 ms, the achieved average response time can be easily calculated using the timestamps. These bilateral results are combined

using the dependency models and impact analysis done at development phase. As a result not only bilateral SLOs are assessed but a complete picture on how the composite service, with the different SLAs, is functioning is provided.

This combined result of development phase model analysis and runtime event log data is graphically represented by *coloring* the services in the dependency models (cf. Fig. 5). A service is colored *green* when the constraint of the SLO is met during runtime, *yellow* when the constraint is not met with a maximum negative deviation of 10%, *red* when the constraint is not met with a deviation greater than 10%, and *dark green* if the constraint was easily met (i.e., the constraint could have been more than 10% tighter). Of course, these deviation percentages can be changed by the user. An important aspect of this evaluation is the *time frame* as defined in the SLO. The SLO time frame of the composite service might differ from the time frame defined in the SLOs of the services it depends on. For example, the cost SLO for SubscribedNews with the customer evaluates an average of *one month* while the time frame for the two services it depends on is a maximum *per day* (cf. Table 1). Since we aim at monitoring the composite service, we choose to evaluate each SLO with the time frame used in the composite service. This means that for evaluating the composite service some SLOs of services it depends on, have to be adapted. In case of the cost SLO for SubscribedNews this means that the costs of UpdateNews services are gathered for the entire month. Now, the constraint of "not more than 40 euro per day" (cf. Table 1) is evaluated by checking whether there was no violation during that month.

Estimations are also evaluated. For example, the company estimated that in 75% of all cases CP1 would be the first to deliver content to a request of data (cf. Sect. 4). This is represented as a ratio on the XOR-split. These estimations on constructs are evaluated based on combining information from different bilateral monitoring results. The results are again graphically represented by coloring the constructs.

As a last addition the *impact factors*, as calculated in Sect. 4, are considered. The result of these three parts (evaluating the SLOs, estimations on constructs, and impact factors) combined form a solid basis to evaluate the composite service. As an example, the response time SLO of NewsRequest is evaluated and graphically represented in (cf. Fig. 5). Using the event log model, it is calculated that the average response time to a NewsRequest is 5.2 ms, this is less than 10% under performance and therefore colored yellow. It turns out that ratio r:s (cf. Sect. 4) is not 3:1 but 4:1. CP1 was more often favored over CP2 than expected. As a result the upper half of the XOR-join is green and the lower half yellow. The response time by CP1 was on average 3.4 ms which is more than 10% worse than agreed upon. As a result the service colors red. CP2 responded

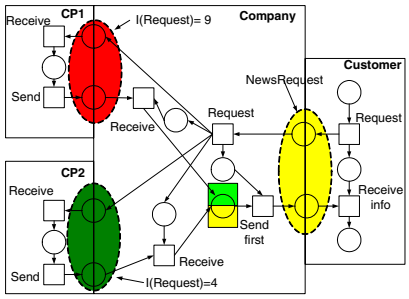


Figure 5. Coloring of a Dependency Model

on average in 3.6 ms which is more than 10% better than agreed. Therefore, the service colors dark green.

The result of our approach enables users to identify which services are performing good and which bad. In this case a possible cause of violating the response time SLO for NewsRequest is bad performance of CP1. Especially considering that the impact factor of this service is more than twice as big as the impact factor of the other NewsRequest service (9 against 4). Also estimations on choices, repetitions, and reuse are evaluated. In this way, estimations in the dependency models can be compared to real life data, enabling decision support for managing composite services.

6. Related Work

An important *framework* for specifying composite services is the WSLA framework by Keller et al. [12]. We use their structure and requirements on SLA definitions for our approach. Their framework enables the specification of SLAs which enables monitoring composite services although this is not treated explicitly. Another framework for specifying web services is the WSOL framework by Tosic et al. [15]. Unique in this approach is that it enables offering services in *classes* rather than per instance. Tosic et al. also treat the dependencies between different services in [9].

Sahai et al. [14] aim at automated SLA *monitoring* by specifying SLAs and not only considering provider side guarantees but focus also on distributed monitoring, taking the client side into account. Barbon et al. [3] enable runtime monitoring while separating the business logic from the monitoring functionality. For each instance of a process a monitor is created. Unique in this approach is the ability to also monitor *classes* of instances, enabling abstraction from an instance level. The smart monitoring approach of Baresi et al. [4] implements the monitor itself as a service. There are three types of monitors available for different aspects of the system. Their approach is developed to monitor specifically contracts with constraints. In [5] Baresi et al. present an approach to dynamically

monitor BPEL processes by adding monitoring rules to the different processes. These rules are executed during runtime. Our approach does not require modifications to the process descriptions what might suit better to some application areas. An interesting approach in this direction is work by Mahbub et al. [13] who, as en exception, do consider the whole state of the system in their monitoring approach. They aim at monitoring derivations of behavior of the system. The requirements for monitoring are specified in event calculus and evaluated with run-time data. Although many of above mentioned approaches do consider third parties and also allow abstraction of results for composite services, none of them addresses how to create this abstraction in detail. Problems like matching messages from different processes as in our SubscribedNews example where databases are used, are not considered.

Another research community analyzes *root causes* in services. In responding approaches dependency models are used to find causes of violations *within* a company. Here, composite services are not considered but merely services the company is responsible for on its own. For example, Agarwall et al. [2] determine the cause of a problem by using dependency graphs. Especially finding the cause of a problem when a service has an SLA with different metrics is here a challenging topic. Also Caswell et al. [8] use dependency models for managing internet services. Again, focussing on finding internal causes for problems. In our work we identify causes of violations in other services rather than internally. Furthermore, our dependencies between different services are on the same level of abstraction while in root cause analysis one service is evaluated on different levels of abstraction.

7. Summary and Outlook

In this paper we describe our MoDe4SLA approach which combines analysis information on dependency relations between services, and their impact factor on the composite service during development phase with bilateral monitoring results during runtime. The result supports the company in managing its composite services for detection and coverage of SLA violations. More specifically, decisions on whether or not to change content provider, whether or not to change certain SLAs, and which SLAs should be reconsidered are supported by depicting dependencies between services.

In the near future we plan to finalize the implementation and evaluation of our approach. Extensive testing on complex composite services is needed. Furthermore, we plan to extend our dependency based approach with handling complex SLAs where also dependencies *between* and *within* different SLOs occur. For example, SLAs where two different SLOs concern response time, or where one SLO concerns not only response time but also cost.

References

[1] Workflow patterns. http://www.workflowpatterns.com.

[2] M. K. Agarwal, K. Appleby, M. Gupta, G. Kar, A. Neogi, and A. Sailer. Problem determination using dependency graphs and run-time behavior models. In *DSOM*, volume 3278, pages 171–182. Springer, 2004.

[3] F. Barbon, P. Traverso, M. Pistore, and M. Trainotti. Run-time monitoring of instances and classes of web service compositions. In *ICWS '06: Proc. of the IEEE Int. Conference on Web Services*, pages 63–71, Washington, DC, USA, 2006. IEEE Computer Society.

[4] L. Baresi, C. Ghezzi, and S. Guinea. Smart monitors for composed services. In *ICSOC '04: Proc. of the Int. Conference on Service Oriented Computing*, pages 193–202, New York, NY, USA, 2004.

[5] L. Baresi and S. Guinea. Towards dynamic monitoring of WS-BPEL processes. In *ICSOC*, volume 3826, pages 269–282. Springer-Verlag, 2005.

[6] L. Bodenstaff, A. Wombacher, and M. Reichert. On formal consistency between value and coordination models. Technical Report TR-CTIT-07-91, Univ. of Twente.

[7] J. Cardoso, A. Sheth, J. Miller, J. Arnold, and K. Kochut. Quality of service for workflows and web service processes. *Web Semantics: Science, Services and Agents on the World Wide Web*, 1(3):281–308, 2004.

[8] D. Caswell and S. Ramanathan. Using service models for management of internet services. *IEEE Journal on Selected Areas in Communications*, 18(5):686–701, 2000.

[9] B. Esfandiari and V. Tosic. Towards a web service composition management framework. In *ICWS '05: Proc. of the IEEE Int. Conference on Web Services*, pages 419–426, Washington, DC, USA, 2005. IEEE Computer Society.

[10] M. C. Jaeger, G. Rojec-Goldmann, and G. Muhl. QoS aggregation in web service compositions. In *Proc. of the Int. Conference on e-Technology, e-Commerce and e-Service*, pages 181–185, Washington, DC, USA, 2005. IEEE Computer Society.

[11] K. Jensen. *Coloured Petri Nets. Basic Concepts, Analysis Methods and Practical Use*. Springer, 1997. Three Volumes.

[12] A. Keller and H. Ludwig. The WSLA framework: Specifying and monitoring service level agreements for web services. *Journal of Network and Systems Management*, 11(1):57–81, 2003.

[13] K. Mahbub and G. Spanoudakis. Run-time monitoring of requirements for systems composed of web-services: Initial implementation and evaluation experience. In *Proc. of the Int. Conference on Web Services*, pages 257–265. IEEE Computer Society, 2005.

[14] A. Sahai, V. Machiraju, M. Sayal, A. P. A. van Moorsel, and F. Casati. Automated SLA monitoring for web services. In *DSOM '02: Proc. of the 13th IFIP/IEEE Int. Workshop on Distributed Systems: Operations and Management*, pages 28–41, London, UK, 2002. Springer-Verlag.

[15] V. Tosic, B. Pagurek, K. Patel, B. Esfandiari, and W. Ma. Management applications of the web service offerings language (WSOL). *Information Systems*, 30(7):564–586, 2005.